Tutorial de uso de HAPTIK bajo Windows

Este tutorial ha sido generado gracias al trabajo de María Luisa Pinto Salamanca, de la Universidad Nacional de Colombia

Resumen. Se distinguen dos partes:

- una primera donde se explica la instalación bajo windows de la librería y el desarrollo de aplicaciones en C++ y bajo Matlab,
- Se presentan dos ejemplos de aplicación de la librería **haptik** con el lenguaje de programación C++ y una implementación a mayor nivel con Matlab & Simulink. Se indican los pasos de descarga, instalación y desarrollo de proyectos para cada caso.

Prerrequisitos

se requiere la instalación previa de Matlab & Simulink y un programa de compilación de C++, en este caso, los ejemplos fueron desarrollados con Microsoft Visual Studio 2005.

Parte 1. Introducción a HAPTIK

HAPTIK es una librería de código abierto que puede ser descargada de la página web http://www.haptiklibrary.org/ . Fue desarrollada por Investigadores del Siena Robotics and Systems Lab. Italia. Provee una capa de abstracción de hardware para acceder a dispositivos hapticos de diferentes fabricantes fácilmente, permitiendo desarrollar aplicaciones independientes de configuraciones de APIs, hardware o controladores particulares.

No esta vinculada a ninguna librería grafica ni a un SDK especifico para la detección de colisiones. Ha sido diseñado para ser integrado a través de programación con clases, permite accesos basados en Callbacks y puede usarse en diferentes sistemas de renderización grafica (OpenGL o DirectX).

HaptiK consiste en una recarga dinámica de plugins de ejecución que pueden ser fácilmente ampliados o personalizados no solamente a partir del lenguaje de programación C++ sino también Matlab® y Java™.

Arquitectura

Haptik library se pueden analizar desde dos puntos de vista, el manejo directo de librerías dinámicas accedidas en tiempos diferentes o con respecto a una subdivisión de componentes e interfaces acezadas simultáneamente.

Si se utilizan DLLs (Dynamic Link Libraries implementadas en GNU/Linux como Shared Objects), se deben tener en cuenta tres niveles de acceso: desde la aplicación, la librería o los plugins (ver figura 1). HAPTIK.LIBRARY.DLL enlaza el nivel de aplicación con Haptik library. Los plugins son un conjunto de DLLs que no ven directamente la aplicación pero si contienen las

funcionalidades requeridas para usar un conjunto de dispositivos hapticos. La librería actúa como un manejador: carga plugins cuando son necesarios, hace que estos sean leídos y actualizados y garantiza que la información se entregue al nivel de aplicación. Lo cual en tiempo de ejecución ofrece dos ventajas:

- La aplicación no depende de la presencia de un controlador dll para un dispositivo específico, sino solamente para aquellos que sean cargados a través de los plugins.
- Compatibilidad binaria para nuevos dispositivos a través de la simple adición del plugins para el nuevo dispositivo soportado.

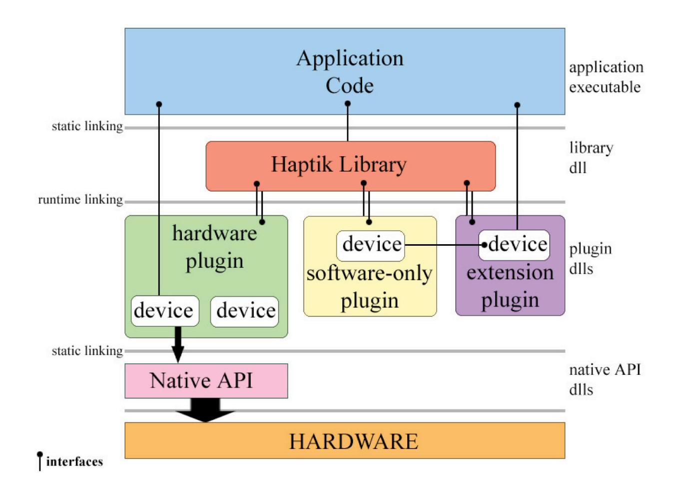


Figura 1. Arquitectura de Haptik Library [9].

Todas las librerías y plugins de cada dispositivo son implementados como componentes y todos los canales de comunicación (aplicación-librería, librería-plugins, aplicación-dispositivo) están basados en un interface. De esta forma la aplicación puede acceder a todos los dispositivos de forma uniforme, permitiendo una misma ejecución con hardware de diferentes fabricantes.

Descarga, Instalación y desarrollo de Aplicaciones:

- En la página http://sirslab.dii.unisi.it/haptiklibrary/download.htm se ofrece la librería a través de una archivo de ejecución directa, las distribuciones del código fuente y la documentación respectiva. La últimas versiones disponibles HAPTIK LIBRARY 1.1RC1 y Haptik.Library.1.1rc1.sources.IA32.20070107.zip.

- En el Manual de Usuario incluido en el archivo de instalación se presenta una descripción detallada para el desarrollo de aplicaciones con la librería HaptiK (Chapter 4 Application Development), la cual incluye desde una librería dll (Haptik.Library.dll), una serie de plugins (Haptik.PluginName.dll), algunos archivos de cabecera (Haptik.hpp) y un archivo .lib (Haptik.Library.lib) construido para compiladores de VisualC++ (para usar otros compiladores deberá construirse nuevamente el proyecto de Haptik).

- Los pasos básicos para una aplicación con VisualStudio 8.0 se resumen en la figura 2. Inicialmente se deberán adicionar al proyecto los directorios include y lib de Haptik. Los

plugins dlls podrán ubicarse en system/user path para que sean accesibles por la aplicación. Usualmente, solo basta con adicionar el directorio /RSLib/bin en el sistema PATH.

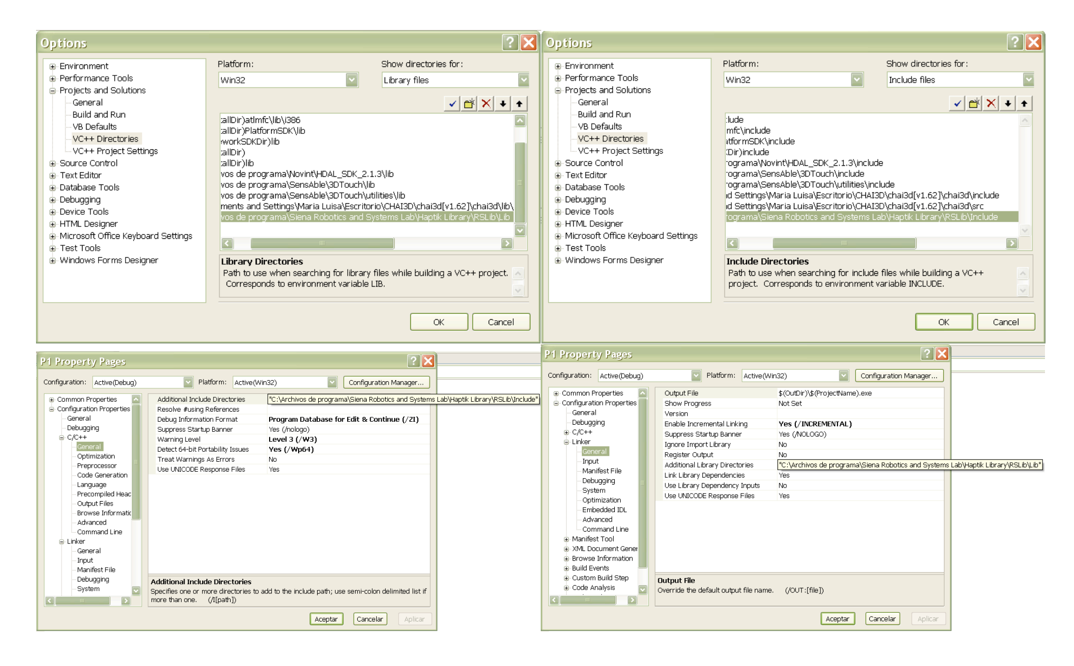


Figura 2. Adición de directorios include y Lib de HaptiK.

- En la aplicación deberá incluirse además la cabecera RSLib\Haptik.hpp acompañada de RSLib namespace, para evitar conflictos de nombres. A continuación un sencillo ejemplo de la aplicación inicial en la que se recocnoce automáticamente el dispositivo conectado y se aplica un vector de fuerzas con F(x,y,z), con las componente x y z = 0.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <RSLib\Haptik.hpp>
using namespace RSLib;
using namespace RSLib::Math;
//---DEFINES
// Desc: undef this to use procedure callback
//------------------------------------------->
//#define PROCEDURAL_CALLBACK


void Callback(HaptikData& data)
{
   data.forceFeedback = data.position.y < 0 ?
                        Vector3(0,-data.position.y,0) :
                        Vector3(0,0,0);
}

void main()
{
   IHaptikDeviceInterface device;
   device = Haptik().GetHaptikDeviceInterface();
   device->Init( Callback );
   device->Start();

   while(!kbhit());

   device->Release();
}
```




- El método GetHaptikDeviceInterface puede ser accedido a través de una interfaz que facilita su manejo, se trata de la clase Haptik que permite una carga directa de las librerías dll de cada dispositivo. Con esto, la función principal del ejemplo anterior podría ser:

```
void main()
{
Haptik haptik;
IHaptikDeviceInterface device;
device = (IHaptikDeviceInterface) haptik.GetDeviceInterface();
device->Init(Callback);
device->Start();
getch();
device->Release();
}
```

- La lectura de estado y aplicación de señales de fuerza y torque (si el dispositivo lo permite) se simplifican notablemente, al reducir el programa al manejo de funciones tipo callback. Se deben tener en cuenta para la visualización en escena, las transformaciones y el sistema coordenado, de acuerdo a las librerías de renderizado a utilizar.

```
void Callback(HaptikData& data)
{
   printf("POS:(%0.3f,%0.3f,%0.3f)\n",data.position.x,data.position.y,data.position.z);
   Posicion = data.position;
   // PARA TRANFORMACION DE COOREDNADAS FISICAS A MODELVIEW
   Vector3 Posicion;
   world = data.matrix;
   FLT32 rx = RotationAround(Vector3(0,1,0), data.matrix.r1.xyz,Vector3(1,0,0));
   FLT32 ry = RotationAround(Vector3(1,0,0), data.matrix.r0.xyz,Vector3(0,1,0));
   FLT32 rz = RotationAround(Vector3(1,0,0), data.matrix.r0.xyz,Vector3(0,0,1));

   data.forceFeedback = data.position.y < 0 ?
                        Vector3(0,-data.position.y,0) :
                        Vector3(0,0,0);
printf("POSDisp:(%0.3f,%0.3f,%0.3f)\n",PosicionDisp[0],PosicionDisp[1],PosicionDisp[2]);
}
```

Aplicaciones con HaptikLibrary, Matlab & Simulink

- Las aplicaciones de Matlab con Haptik Library se pueden realizar a través de líneas de comando y archivos .m cuya única restricción es que tengan acceso a la librería "haptik_matlab.dll", haptikdevice_list y la carpeta @haptikdevice. haptikdevice_list es una función que retorna los dispositivos conectados; @haptikdevice contiene una clase de utilidades para simplificar el manejo de haptik_matlab.dll.

- Las funciones principales de esta librería son:

  - Haptikdevice_list: imprime el listado de los dispositivos hapticos conectados.
  - h = haptikdevice, obtiene la información del dispositivo por defecto.
  - h = haptikdevice(id), carga un dispositivo especifico
  - h = haptikdevice(@mycallback,rate), abre el dispositivo por defecto y usa la función mycallback(.m) para especificar la frecuencia del hilo haptico.
  - h = haptikdevice(id,@mycallback,rate), abre un dispositivo especifico y usa la

función mycallback(.m) para especificar la frecuencia del hilo haptico.

- [matrix,button] = read(h), obtiene la orientación del dispositivo (matrix(4,1) matrix(4,2) and matrix(4,3) contains position).
- position = read position(h), lee la posicion del dispositivo.
- button = read button(h), lee el estado de los pulsadores del dispositivo.
- write(h,ff), envia fuerzas y torques al dispositivo. FF puede ser 2x3, 3x2, 1x3, 3x1, 1x6, 6x1.
- close(h), cierra la aplicación del dispositivo.

▪ Las funciones callback se implementan con temporizadores basados en Java, por lo que no se garantiza completa sincronización con el hilo haptico. La figura 7 presenta un ejemplo de ejecución y lectura de posición del dispositivo Phantom Omni desde la línea de comando de Matlab.

▪ Para las aplicaciones con Simulink, HaptiK Libray se incluye además unas funciones S-function en la librería haptik simulink.dll que determinan la información del dispositivo y el tiempo de ejecución de manera que pueda sincronizarse con el hilo haptico. La figura 3 muestra en ejemplo de ejecución con el Phantom Omni. Si no hay dispositivos conectados, el plugin Spectre Mause permite simular el comportamiento haptico de igual manera que con el resto de aplicaciones en C++.

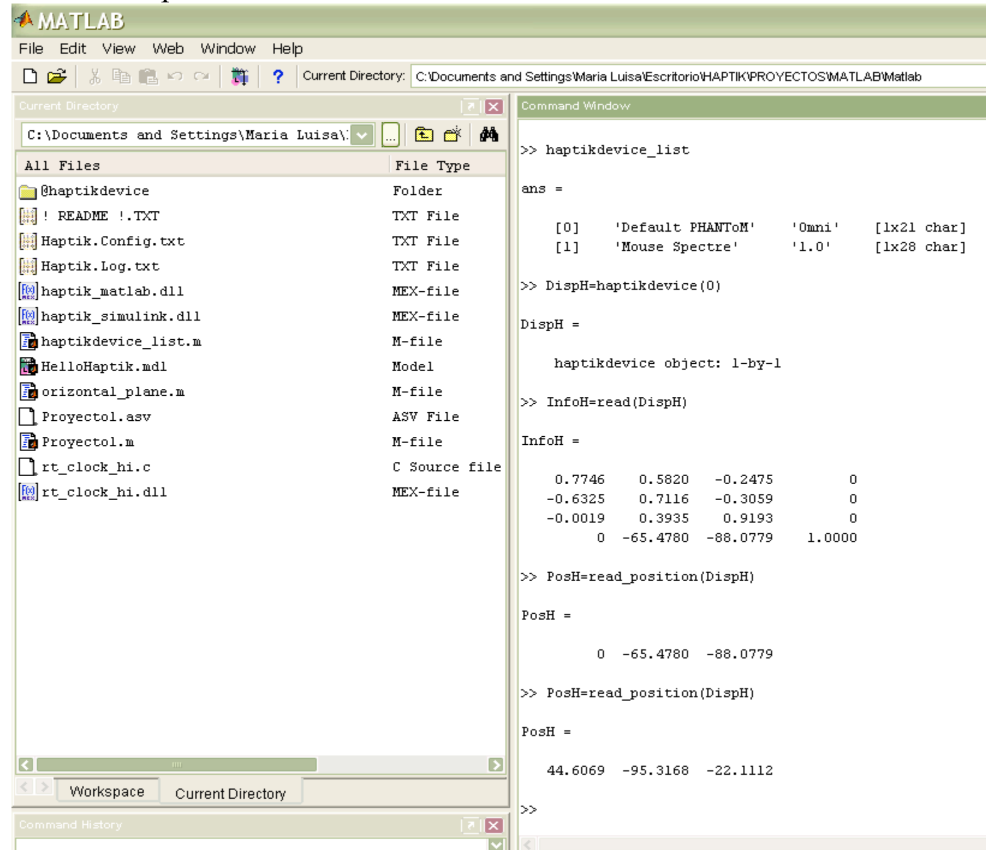


Figura 2. Ejemplo de ejecución y lectura de posición del dispositivo Phantom Omni con Matlab.

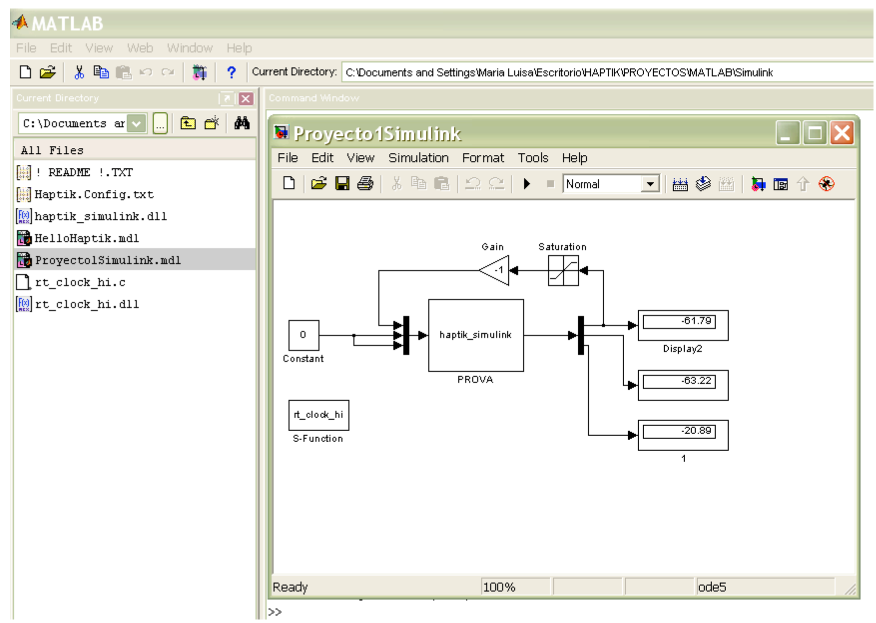


Figura 3. Ejemplo de ejecución y lectura de posición del dispositivo Phantom Omni con Simulink

Parte 2. ejemplos de aplicaciones básicas con HAPTIK

Aplicaciones de Haptik Library con Microsoft Visual Studio 2005:

- En el Manual de Usuario incluido en el archivo de instalación se presenta una descripción detallada para el desarrollo de aplicaciones con la librería HaptiK (Chapter 4 Application Development), incluyendo una librería dll (Haptik.Library.dll), una serie de plugins (Haptik.PluginName.dll), algunos archivos de cabecera (Haptik.hpp) y un archivo .lib (Haptik.Library.lib) construido para compiladores de VisualC++ (para usar otros compiladores deberá construirse nuevamente el proyecto de Haptik).

- En las opciones de creación de un nuevo proyecto con MSVisualC++ ejemplo (ir File→ New Project → Win32 Console Application → Empty Project), adicionar los directorios include y lib de Haptik, para esto ir a Tools→ options → Projects and Solutions → VC++ Directories → agregar los directorios

C:\Archivos de programa\Siena Robotics and Systems Lab\Haptik Library\RSLib\Lib C:\Archivos de programa\Siena Robotics and Systems Lab\Haptik Library\RSLib\Include

- Los plugins dlls podrán ubicarse en system/user path para que sean accesibles por la aplicación. Usualmente, solo basta con adicionar el directorio /RSLib/bin en el sistema PATH.
- Adicionar en las propiedades del proyecto el directorio include: Nombre Property Pages → C/C++ → General → Additional Include Directories → "C:\Archivos de programa\Siena Robotics and Systems Lab\Haptik Library\RSLib\Include"

- Incluir las librerias de enlace: Nombre Property Pages → Linker → General → Additional Library Directories → "C:\Archivos de programa\Siena Robotics and Systems Lab\Haptik Library\RSLib\Lib".



Aplicación 1:

- En el programa principal incluir además la cabecera RSLib\Haptik.hpp acompañada de RSLib namespace, para evitar conflictos de nombres.

```
#include <windows.h>
#include <conio.h>
#include <RSLib\Haptik.hpp>
using namespace RSLib;
```

- Con GetHaptikDeviceInterface se puede detectar el tipo de dispositivo háptico con el que se esta trabajando. Este método puede ser accedido a través de la interfaz Haptik que facilita su manejo, se trata de una clase que permite una carga directa de las librerías dll de cada dispositivo. La función principal simplemente empezaría como:

```
void main()
{
Haptik haptik;
IHaptikDeviceInterface device;
    device = (IHaptikDeviceInterface) haptik.GetDeviceInterface();
```

- La lectura de estado y aplicación de señales de fuerza y torque (si el dispositivo lo permite) se simplifican notablemente, al reducir el programa al manejo de funciones tipo callback. Para esta aplicación por ejemplo incluir una callback para la lectura de posición del dispositivo haptico por ejes y la aplicación del vector unitario de fuerza: F(x,y,z)=(0,posy,0)/|posy| si posy<0. Note la definición de `Vector3`:

```
void Callback(HaptikData& data)
{
   data.forceFeedback = data.position.y < 0 ?
                        Vector3(0,-data.position.y,0) :
                        Vector3(0,0,0);
}
```

- Finalmente, agregar la inicialización y uso de esta callback en la función principal:

```
void main()
{
Haptik haptik;
IHaptikDeviceInterface device;
device = (IHaptikDeviceInterface) haptik.GetDeviceInterface();
device->Init(Callback);
device->Start();
getch();
device->Release();
}
```

- La detección del dispositivo será automática siempre y cuando se encuentre dentro del listado de hapticos soportados por Haptik. Sino hay ningún dispositivo conectado la aplicación se ejecutara con el “Mouse Spectre” con el que se simula el comportamiento de una interfaz háptica. Mueva el ratón con el botón izquierdo presionado sobre la ventana donde se visualizan los vectores de posición, fuerza y torque. Verifique que la fuerza cambie a medida que cambia la posición en y.

- Conecte el dispositivo haptico, corra la aplicación y verifique la aplicación de la fuerza en el eje y.

Aplicación 2, sincronización con un hilo grafico con OpenGL:

- Genere un proyecto con las características de creación, inclusión y adición de librerías de acuerdo a la aplicación anterior.
- Copiar las siguientes líneas en el archivo *.cpp principal:

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <GL/glut.h>
#include <RSLib\Haptik.hpp>
using namespace RSLib;
using namespace RSLib::Math;

void drawGraphics();
void drawCursor();
void drawSolid();

Matrix4x4 world(1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1);
Matrix4x4 view = LookAtLH4x4( Vector3(0,0,10) , Vector3(0,0,0) , Vector3(0,1,0) );
Matrix4x4 proj = PerspectiveFovLH4x4(RS_HALFPIGRECO/2,1,1,2000);
Matrix4x4 viewport(250,0,0,0,0,-250,0,0,0,0,1,0,250,250,0,1);
POINT point[2];
// para la ventana
#define ancho 800
#define alto 700
double PosicionDisp[3];
Vector3 Posicion;

Haptik haptik;
IHaptikDeviceInterface device;


//clockwise rotation looking towards origin along axis of start to reach stop
FLT32 RotationAround(Vector3& start,Vector3& stop,Vector3& axis)
{
   Vector3 c = Cross(start,stop);
   FLT32 s = Dot(c,axis);
   FLT32 angle = Sign(s) * AngleBetweenNormals(start,stop);
   return angle;
}


void Callback(HaptikData& data)
{
   printf("POS:(%0.3f,%0.3f,%0.3f)\n",data.position.x,data.position.y,data.position.z);
  PosicionDisp[0]=data.position.x;
  PosicionDisp[1]=data.position.y;
  PosicionDisp[2]=data.position.z;
   Posicion = data.position;
   // PARA TRANFORMACION DE COOREDNADAS FISICAS A MODELVIEW
   world = data.matrix;
   FLT32 rx = RotationAround(Vector3(0,1,0), data.matrix.r1.xyz,Vector3(1,0,0));
   FLT32 ry = RotationAround(Vector3(1,0,0), data.matrix.r0.xyz,Vector3(0,1,0));
   FLT32 rz = RotationAround(Vector3(1,0,0), data.matrix.r0.xyz,Vector3(0,0,1));

   data.forceFeedback = data.position.y < 0 ?
                        Vector3(0,-data.position.y,0) :
                        Vector3(0,0,0);
}
// drawing callback function
void glutDisplay()
{
    drawGraphics();
    glutSwapBuffers();
}
```

```
void drawGraphics()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glEnable(GL_COLOR_MATERIAL);
glColor3f(0.9, 0.1, 0.1);
drawCursor();
drawSolid();
}


void drawCursor()
{
glPushMatrix();
glTranslatef(PosicionDisp[0],PosicionDisp[1],PosicionDisp[2]);
glutSolidSphere(0.2, 32, 32);
glPopMatrix();
}


void drawSolid()
{
glPushMatrix();
glutSolidSphere(0.4, 32, 32);
glPopMatrix();
}

// reshape function (handle window resize)
void glutReshape(int width, int height)
{
    static const double kPI = 3.1415926535897932384626433832795;
    static const double kFovY = 40;
    double nearDist, farDist, aspect;
    glViewport(0, 0, width, height);
    nearDist = 1.0 / tan((kFovY / 2.0) * kPI / 180.0);
    farDist = nearDist + 2.0;
    aspect = (double) width / height;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(kFovY, aspect, nearDist, farDist);
    /* Place the camera down the Z axis looking at the origin */
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0, 0, nearDist + 2.0,
              0, 0, 0,
              0, 1, 0);
// TRANFORMACION DE COORDENADAS FISICAS A MODELVIEW
    GLdouble modelview[16];
    GLdouble projection[16];
    GLint viewportt[4];
    glGetDoublev(GL_MODELVIEW_MATRIX, modelview);
    glGetDoublev(GL_PROJECTION_MATRIX, projection);
    glGetIntegerv(GL_VIEWPORT, viewportt);
}

// What to do when doing nothing else
void glutIdle()
{    glutPostRedisplay();}


void initScene()
{
//Haptik haptik;
//IHaptikDeviceInterface device;
device = (IHaptikDeviceInterface) haptik.GetDeviceInterface();
if (device == NULL)
   {
     MessageBox(NULL,(LPCWSTR)L"No Device Available!",(LPCWSTR)L"Error",MB_ICONSTOP);
     RELEASE_HAPTIK_INTERFACE(device);
     exit(0);
   }
UINT32 uRes = device->Init(Callback);
```

```
if FAILED(uRes)
   {
      MessageBox(NULL,(LPCWSTR)L"Failed to Init Device!",(LPCWSTR)L"Error",MB_ICONSTOP);
      RELEASE_HAPTIK_INTERFACE(device);
      exit(0);
   }

uRes = device->Start();
 if FAILED(uRes)
   {
      MessageBox(NULL,(LPCWSTR)L"Failed to Start Device!",(LPCWSTR)L"Error",MB_ICONSTOP);
      RELEASE_HAPTIK_INTERFACE(device);
      exit(0);
   }

SetTimer(NULL,0,100,NULL); // tiempo de espera para inicilizacion
}

void exitHandler()
{ device->Release(); }


int main()
{
       glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
       glutInitWindowPosition(100, 0);
       glutInitWindowSize(ancho, alto);
       glutCreateWindow("HOLA Haptik con OpenGL");
       glClearColor(0.3, 0.3, 0.3, 0);
       glutDisplayFunc(glutDisplay);
       glutReshapeFunc(glutReshape);
       glutIdleFunc(glutIdle);
    // Set up handler to make sure teardown is done.
       atexit(exitHandler);
    // Set up the scene with graphics and haptics
       initScene();
       glutMainLoop();
return 0;
}
```

- En la función principal se indica el esquema de inicialización y desarrollo de una escena grafica con OpenGL: creación de ventana; visualización de elementos de la escena (glutDisplay()) representados por dos esferas, una de las cuales se traslada y rota de acuerdo a los movimientos del cursor haptico (drawSolid(), drawCursor()); renderizacion 3D (glutReshape()); e inicio de aplicación (initScene()).

- La detección del dispositivo se realiza en la inicialización de la escena. El calculo de posición y la aplicación del vector de fuerza se hace nuevamente a través de una callback.

- Compile y ejecute el proyecto. Mueva el dispositivo haptico y visualice el movimiento del cursor en la escena grafica.

Aplicaciones de Haptik Library con MatLab & Simulink:

- Abra el programa Matlab. Como directorio de trabajo actual ubique C:\Archivos de programa\Siena Robotics and Systems Lab\Haptik Library\Matlab. Entre los archivos contenidos en dicha direccion se encuentra haptikdevice_list la cual es una función que retorna los dispositivos conectados. La carpeta @haptikdevice contiene una clase de utilidades para simplificar el manejo de haptik_matlab.dll.

- Ejecute desde la ventana de comandos o desde un archivo *.m, la función haptikdevice_list para determinar si el dispositivo haptico que ha conectado al sistema previamente ha sido reconocido por la aplicación con matlab. Nuevamente la aparición por defecto es la del “Spectre Mause”.

- Asigne una variable al dispositivo que quiera manejar del listado y lea sus variables posición, botones presionados y velocidad. Observelos valores

```
function Proyecto1()
%get a device
haptikdevice_list;
h = haptikdevice(0);
pos_h=read_position(h)
boton_h=read_button(h)
vel_h=read_velocity(h)
```

- De igual manera que con la primera aplicación con VisualC++, permita el movimiento del dispositivo háptico en un plano horizontal xy, mediante la ejecución del programa horizontal_plane.m, en el que se aplica el vector de fuerza F(x,y,z)=(0,posy,0)/|posy| si posy<0. Observe la declaración y uso de las funciones write y close para generar la fuerza y finalizar la aplicación liberando el dispositivo.

```
%get a device
h = haptikdevice;

%run simulation for 10 seconds
tic
while toc < 10
    %read probe position
  pos = read_position(h);
    %check for collision and send back force feedback
  if pos(2)<0
    write(h, -1 * [0 pos(2) 0]);
  else
    write(h,[0 0 0]);
  end
 end
close(h);
clear h
```

- Ejecute el archivo HelloHaptik.dll. observe los bloques contenidos para visualizar la posición en los tres ejes.